# A Generic Finite State Machine Framework for the ACNET Control System

Linden Carmichael
FermiLab

# Talk Outline

- Motivation for using Finite State Machines (FSMs).
- Description of the FSM structure.
- Overview of Client-Server model used by the FSM.
- Illustration of the Web tier used for FSM monitoring.
- Detailed look at FSM integration into Electron Cooling Machine Protection.
- Summary of Pros and Cons of the FSM development.

# FERMILAB

# Handling Accelerator tasks

- Operation of an accelerator requires the generation of many complex tasks.
  - Utilize intricate functions and provide comprehensive responses to external stimuli.
  - Are capable of operating reliably and autonomously.
  - Collect accelerator statistics from front ends and store them in database tables.
  - Detect events and create abort logs, quench logs and downtime logs.
  - Regulate beam current in order to compensate for soft faults.

# Accelerator Task Characteristics

- Tasks need to access and integrate various components of the control system.
  - Device readings and settings
  - Database queries and inserts.
  - Access and fire events, etc
- Tasks can be classified by their usage
  - Transient – single usage or of short duration
  - Persistent – constant usage

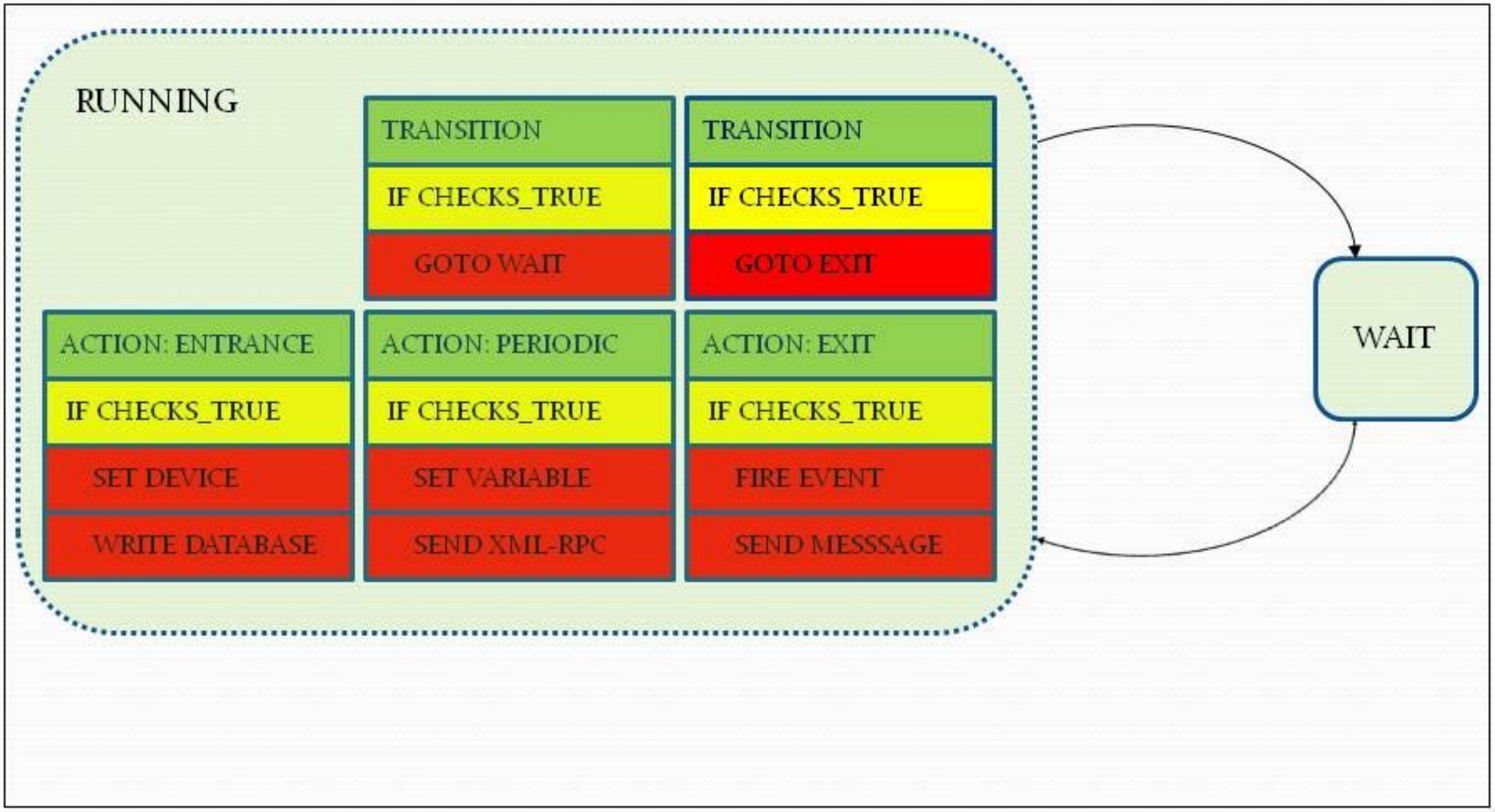
# Finite State Machine

- An event driven Finite State Machine (FSM) framework can realize these tasks.
- Requirements:
  - An intuitive builder capable of resolving complex tasks into their simpler components.
  - Hooks into control system
  - Convenient testing facility
  - Easy deployment
  - Reusability
  - Capacity for persistence

# FSM Structure

- Based loosely upon State Charts
- Event driven FSM
- Single threaded
- Simple and straightforward structure
  - Entrance state
  - Exit state
  - Set of intermediate states with transitions and actions

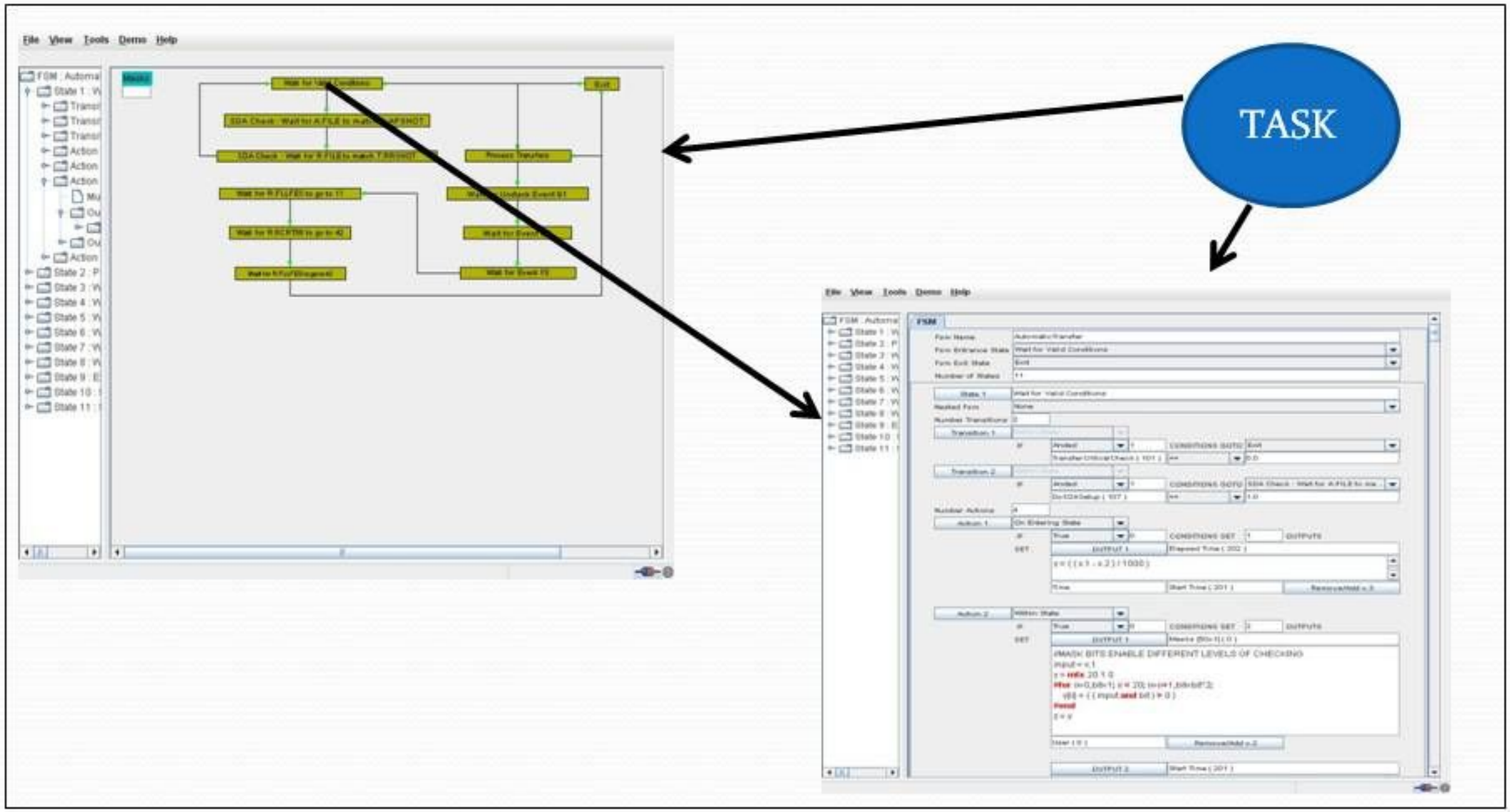
# FSM Structure



# FSM Components

- Hooks into Control System
  - Devices - readings and settings
  - Database - queries and inserts
  - Events - detection and firing
  - ACNET - send messages
  - XML-RPC - remote access
  - ACL - access ACNET Scripting language
- Expression Parser
  - Built-in functions
  - Loops and conditionals
  - Java Reflection

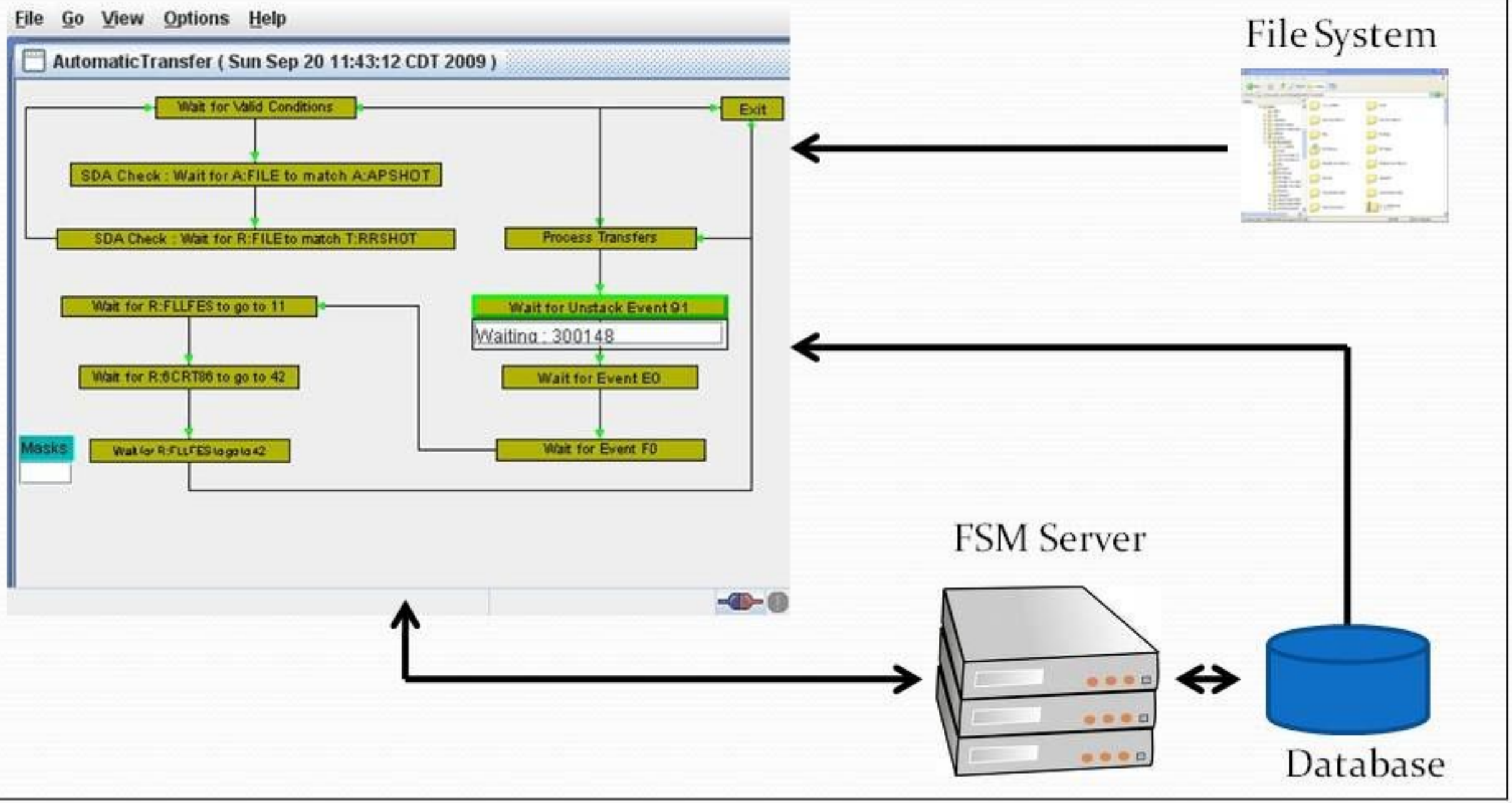
# Building the FSM



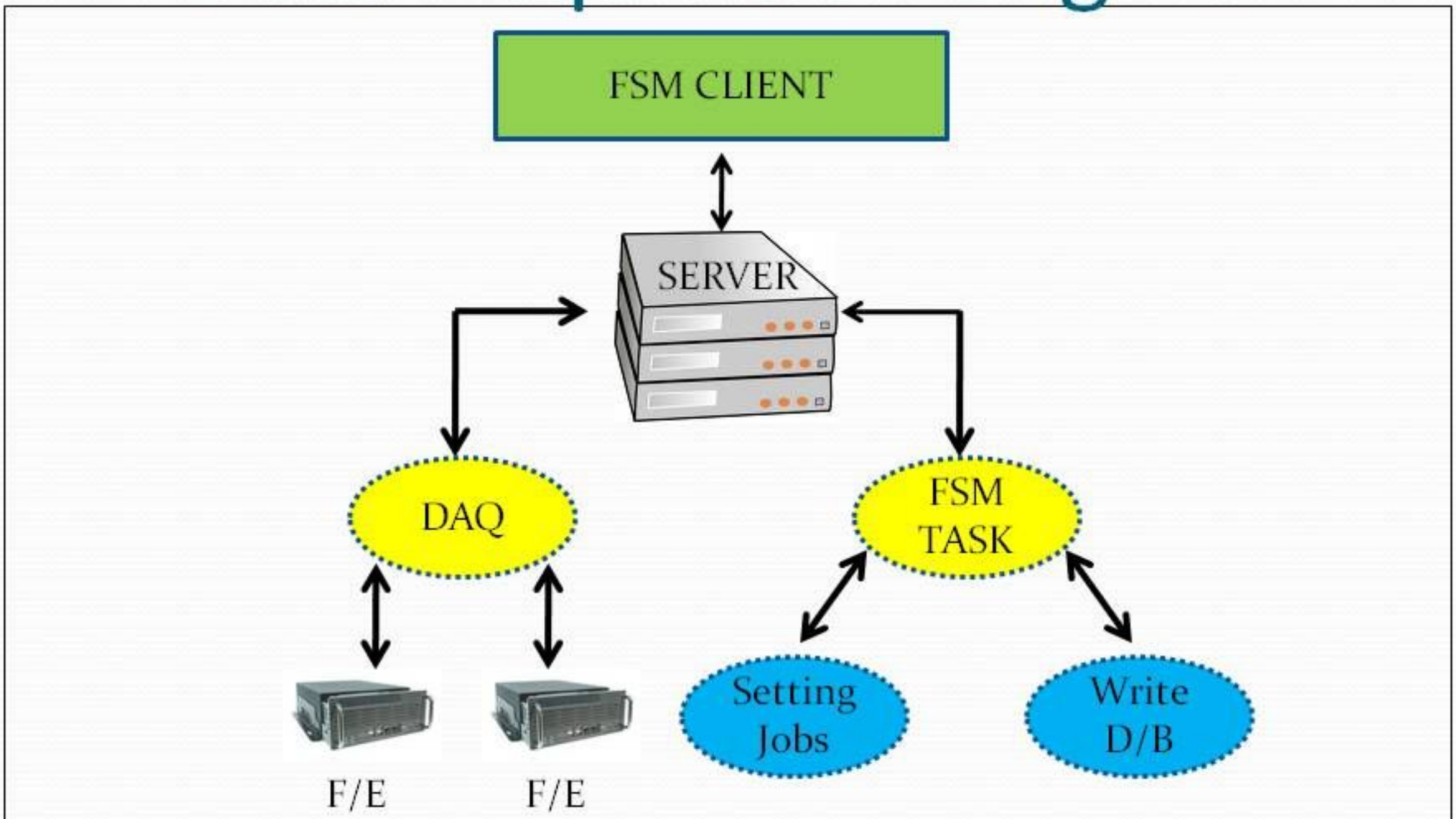
# Client-Server Model

- Clients
  - Thin clients
  - Utilize Java RMI
- Data Acquisition Engine (Server)
  - Java layer to ACNET
  - Servers on high bandwidth nodes
  - Receive and transmit RMI
  - Speak in raw bytes
  - Speak in proprietary ACNET

# Launching the FSM client



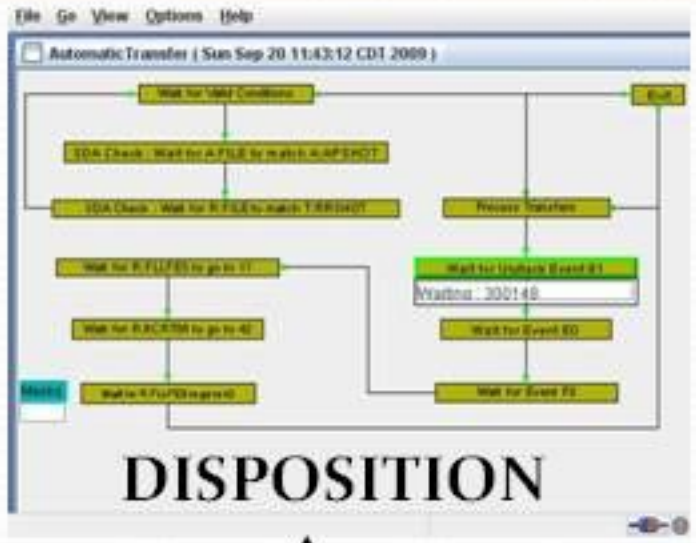
# Data Acquisition Engine

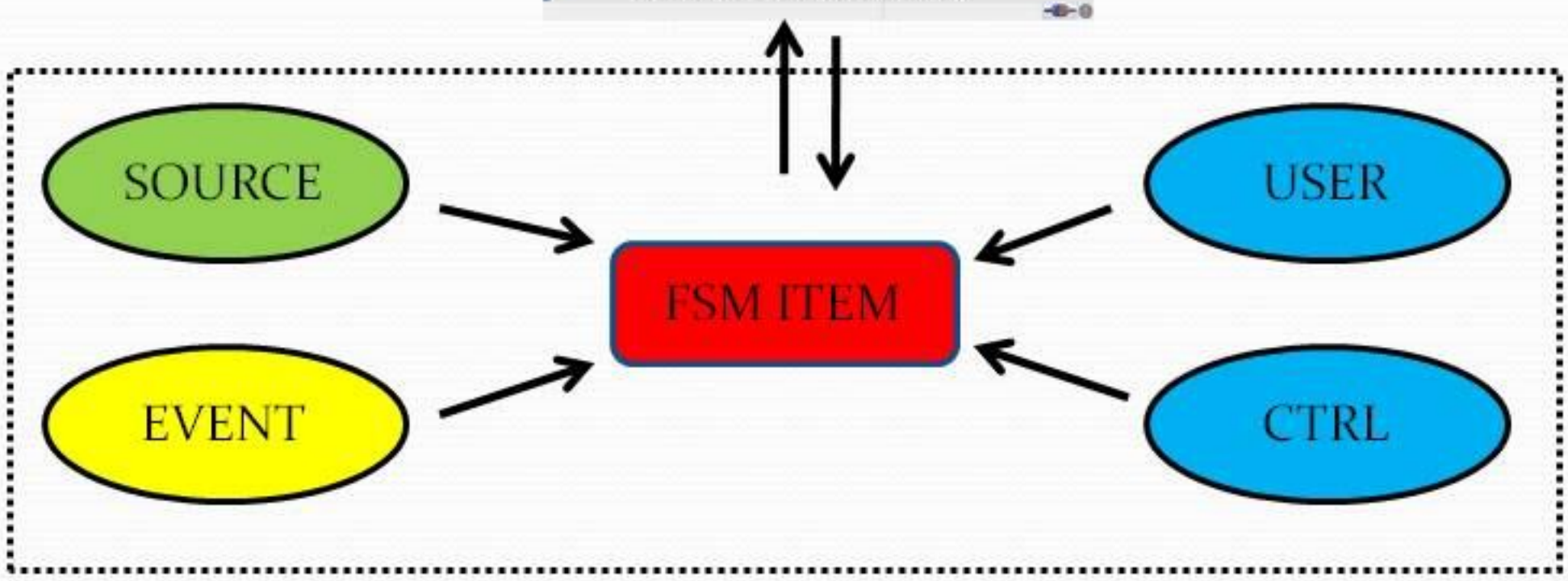


# DAE Jobs

- DAE execution is represented by a 6-tuple job
- User – handles security
- Job control – handles retries, etc
- Source – where data is obtained from
- Event – at what rate to obtain the data
- Item – what to do with the data from the source
- Disposition – where processed data is sent to

# FSM as a DAE job item



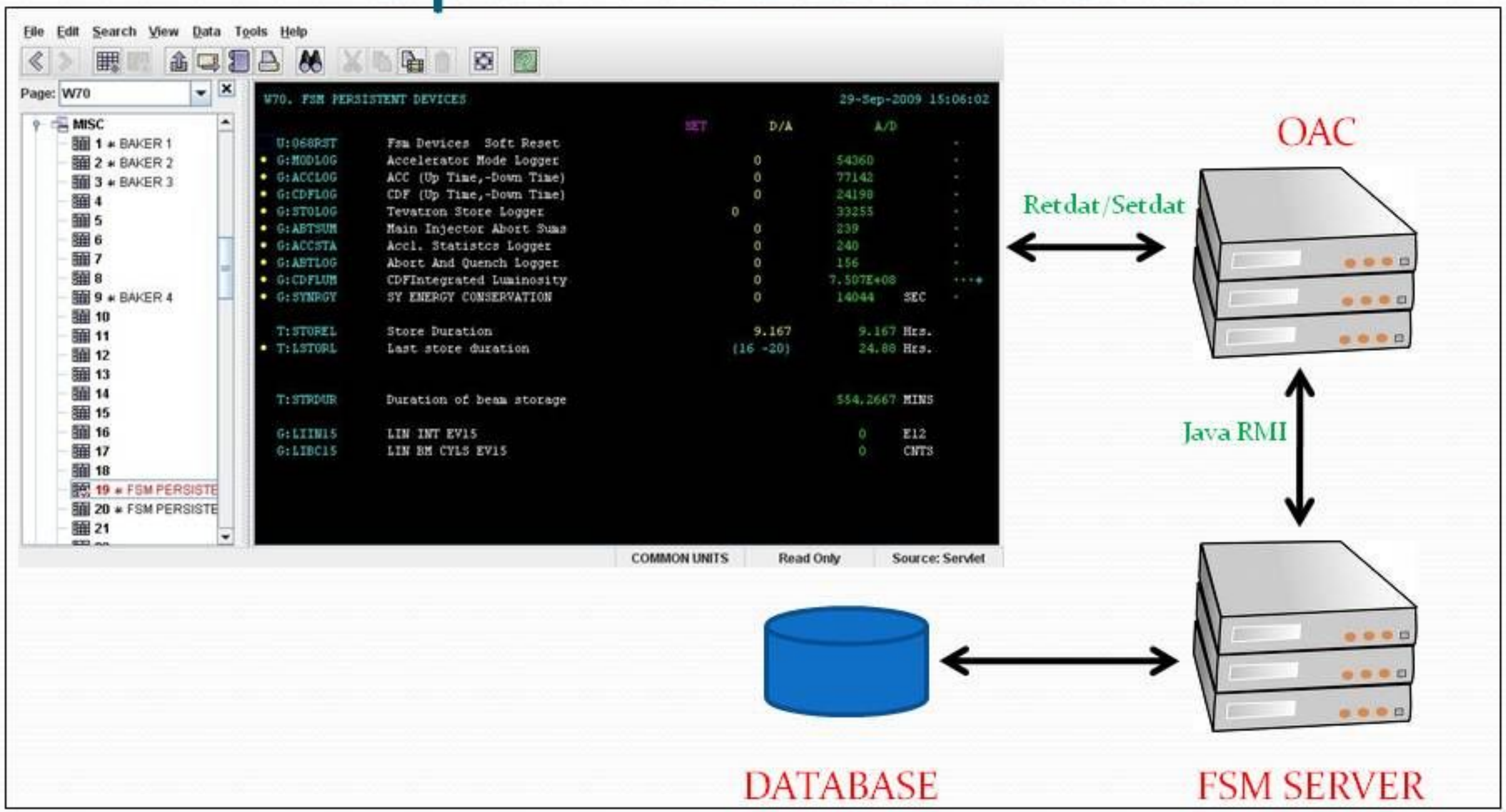
# Persistent FSMs

- FSM jobs owned by an ACNET device.
- Device control bits are used to start and stop jobs.
- Device settings are sent to FSM job.
- Device readings reflect data returned from FSM job.
- Persistent FSMs are supported by the Open Access Client framework.

# Open Access Client (OAC)

- Open Access Client (OAC) is an architecture that allows servers to emulate Front Ends.
- It exists at the ACNET middle tier.
- Provides access to Front End Protocol
  - Alarms
  - Setting Downloads on startup
  - Setting uploads to Database
  - Readings, settings, control, etc

# Open Access Client



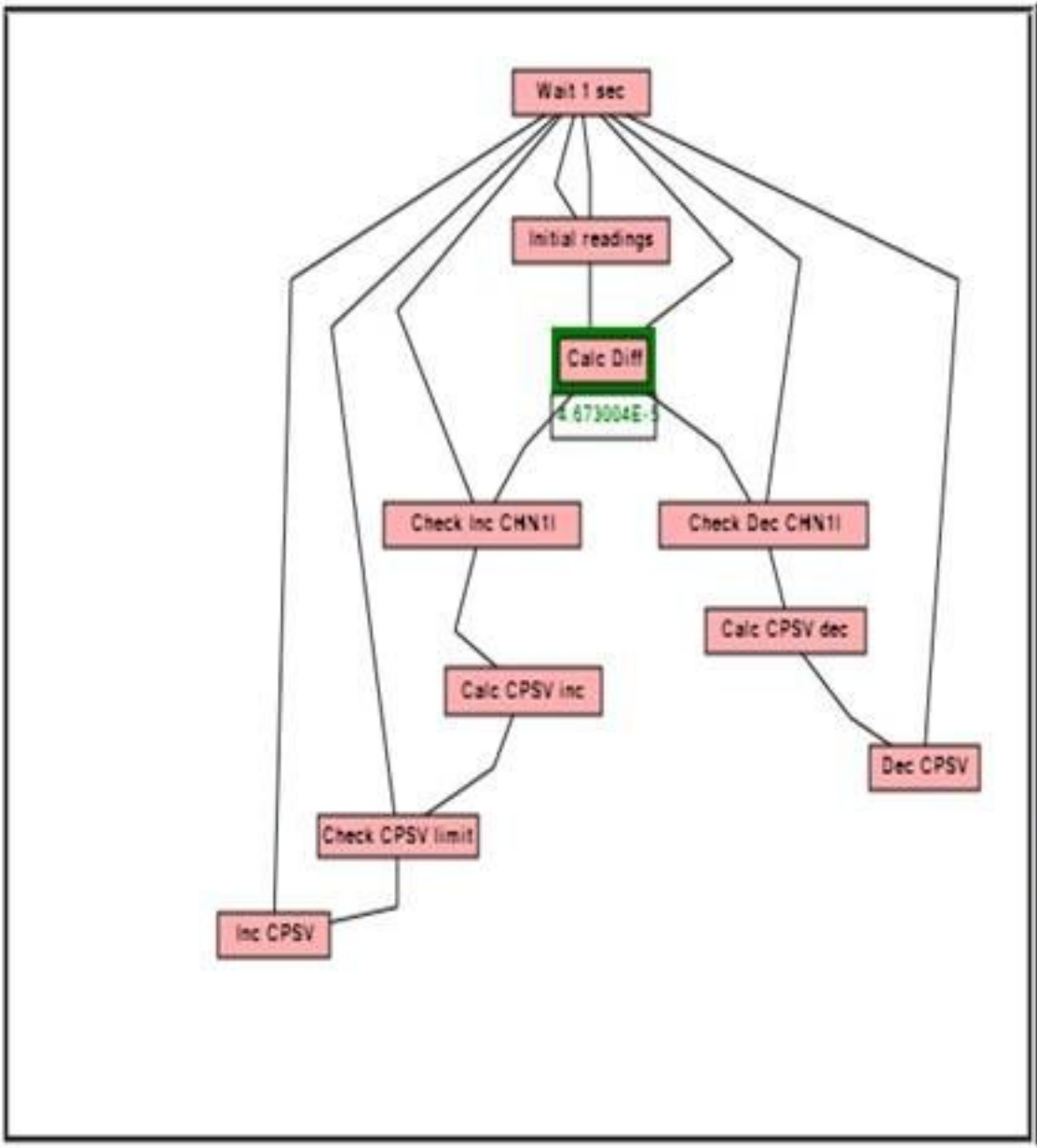
# Web Tier

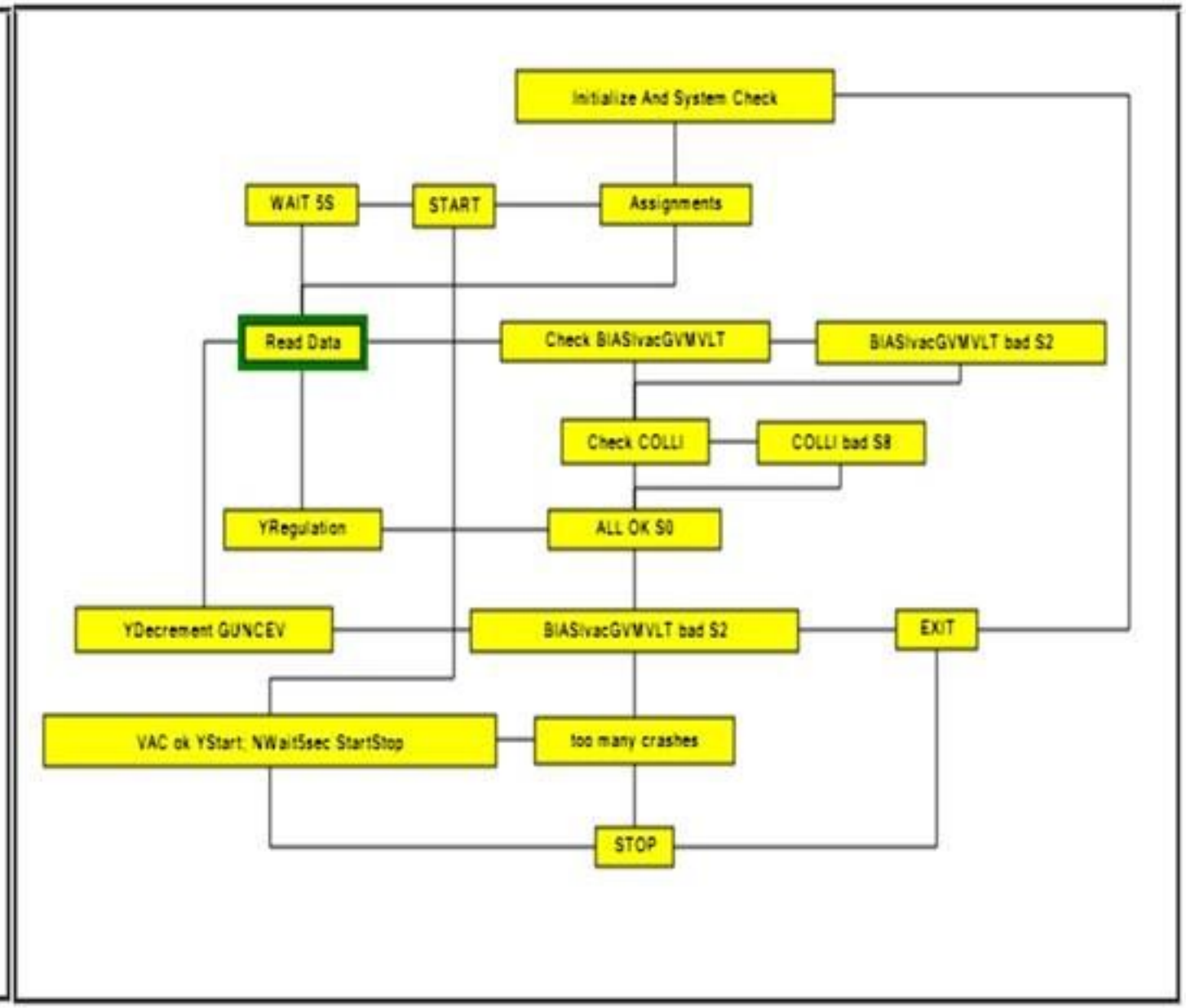
- HTML pages using AJAX and SVG to display FSMs.
- Allows for monitoring of persistent FSMs
- Allows for the generation of summary pages that consists of groups of FSMs and panels.
- FSMs can be started only in safe mode (control disabled)
- Utilizes FSM Views

# FSM Views

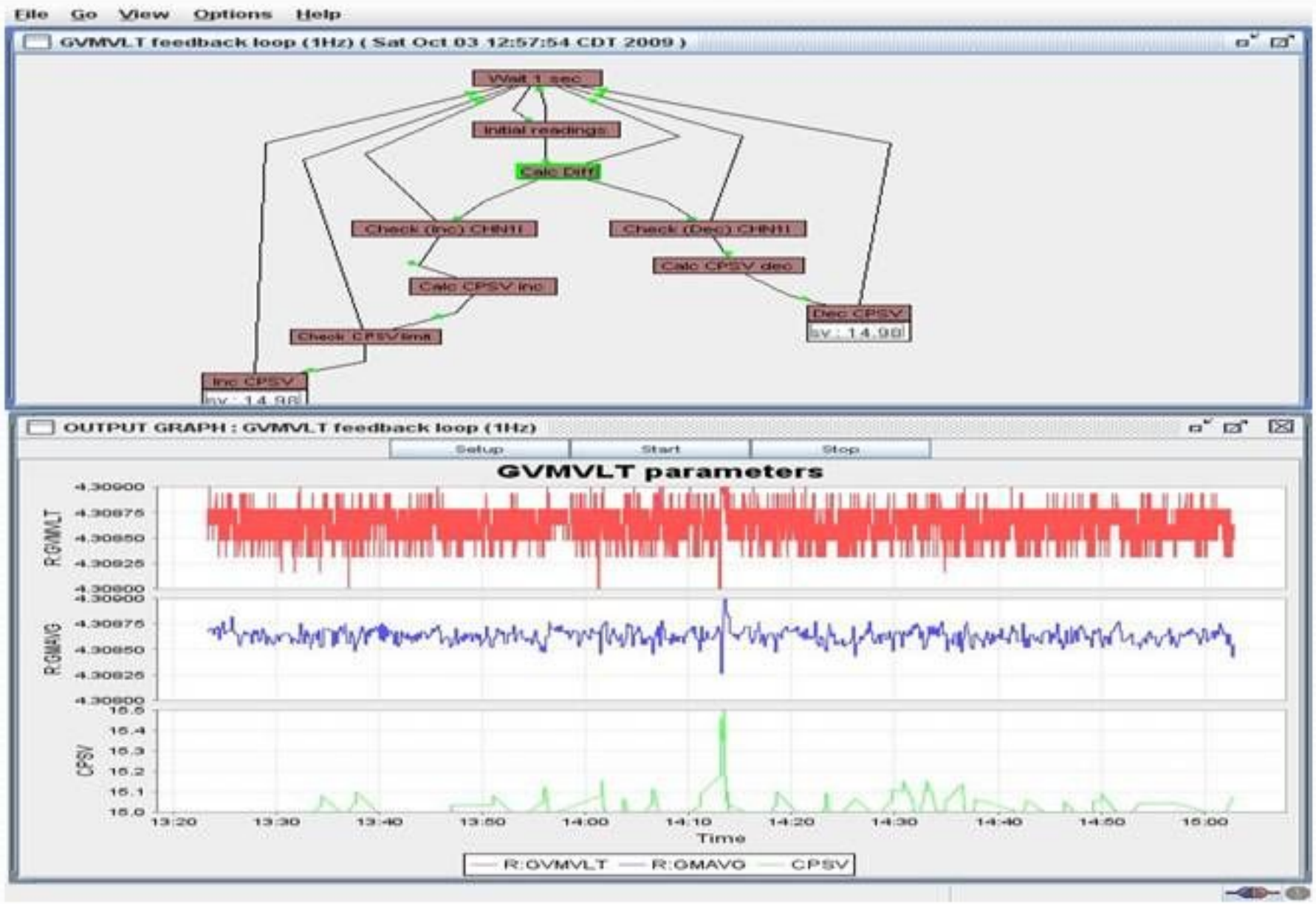
- Extension of thin clients to allow single FSM and multiple views.
- Data returned from FSM server undergoes client-side processing.
- Processing is accomplished with expression parsers and Java reflection.
- User panels and graphs are produced that can provide multiple views of the FSM data returned.

# FSM Summary (Web Page)

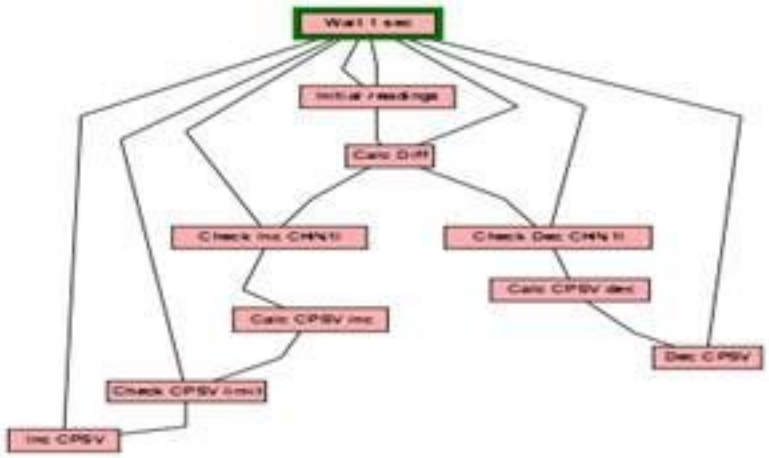


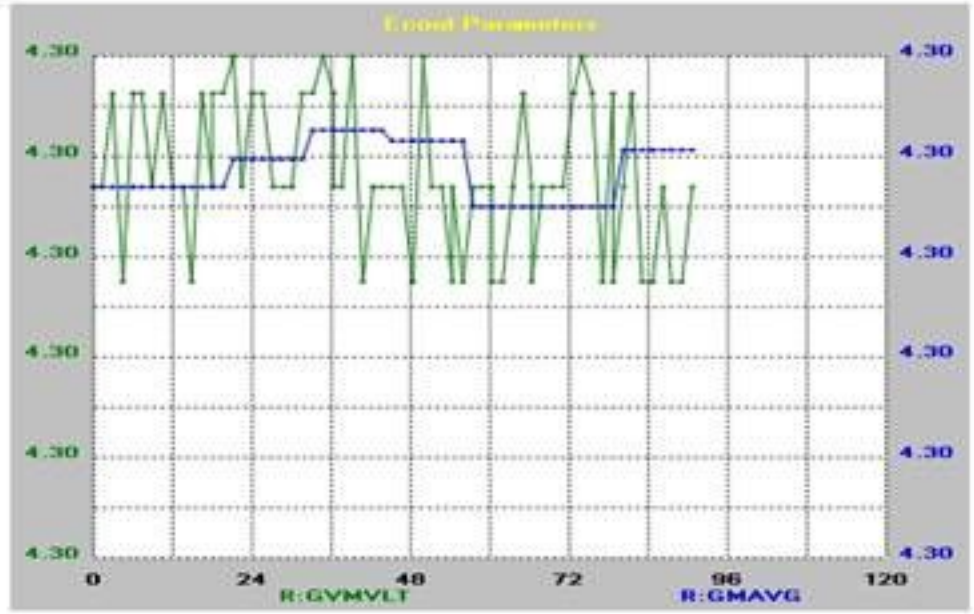



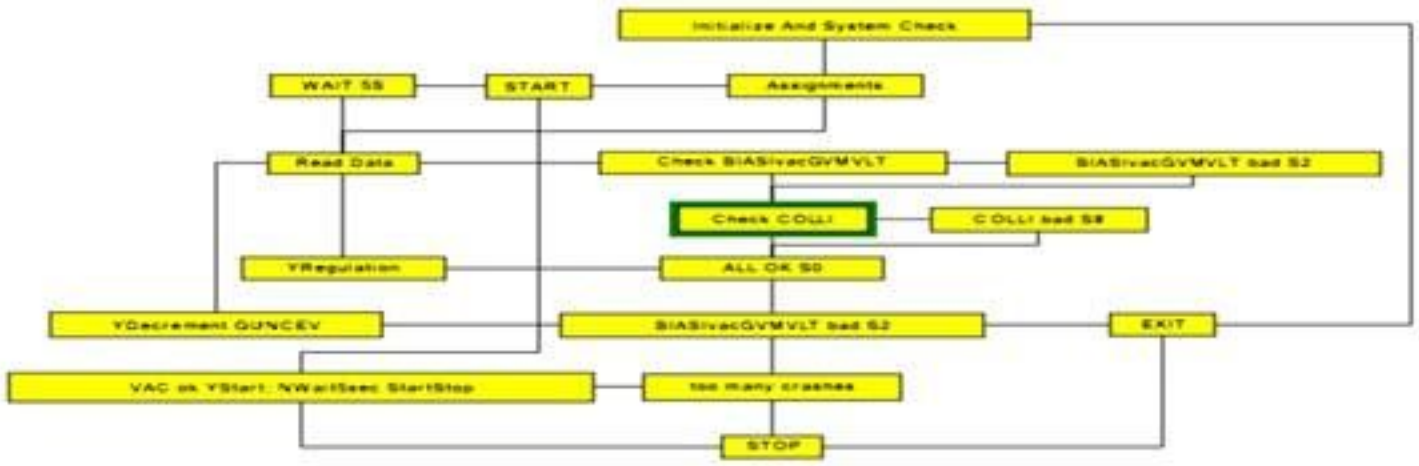
# FSM Views (Application)



# FSM Views (Web Page)



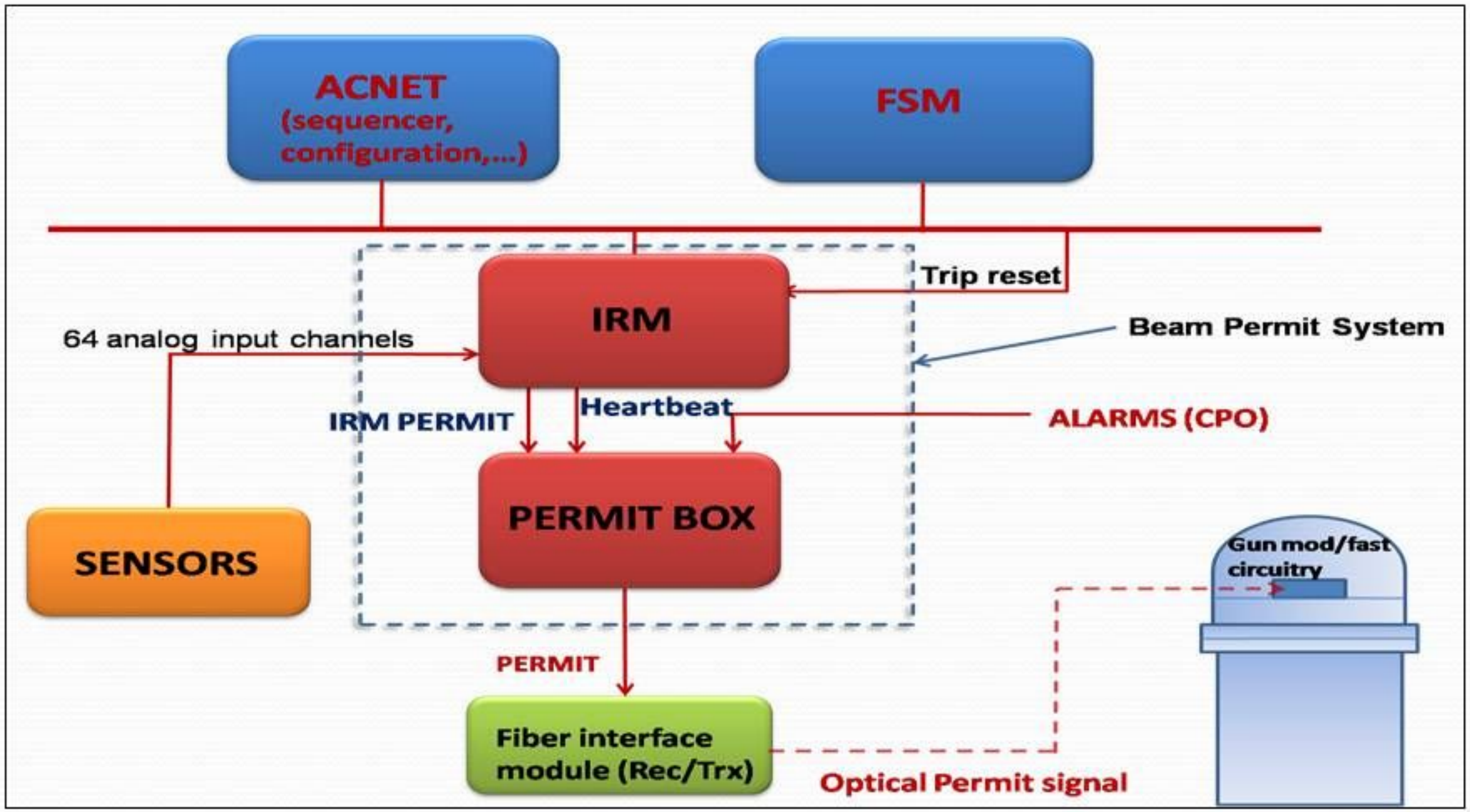



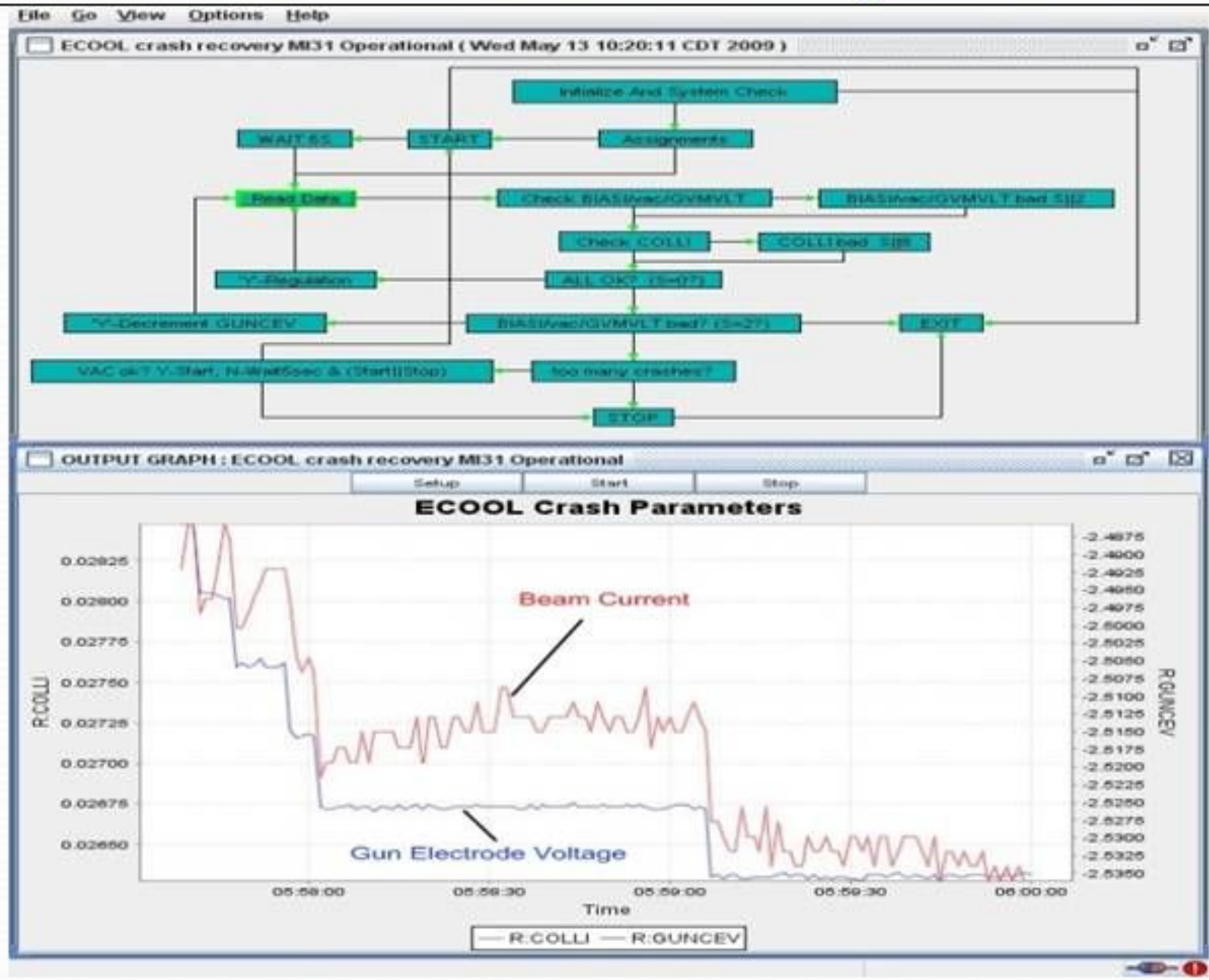
# FSMs in Pelletron's MPS

- Pelletron is a 4.3Mev. 0.1-A, DC electrostatic accelerator used in Electron Cooling.
- The Machine Protection System (MPS) consists of a Permit system and a Regulation system driven by several FSMs.
- Crash Recovery FSM reduces beam current to compensate for soft faults such as vacuum deterioration.
- Slow feedback FSM compensates for drifts in the Pelletron's regulation voltage.

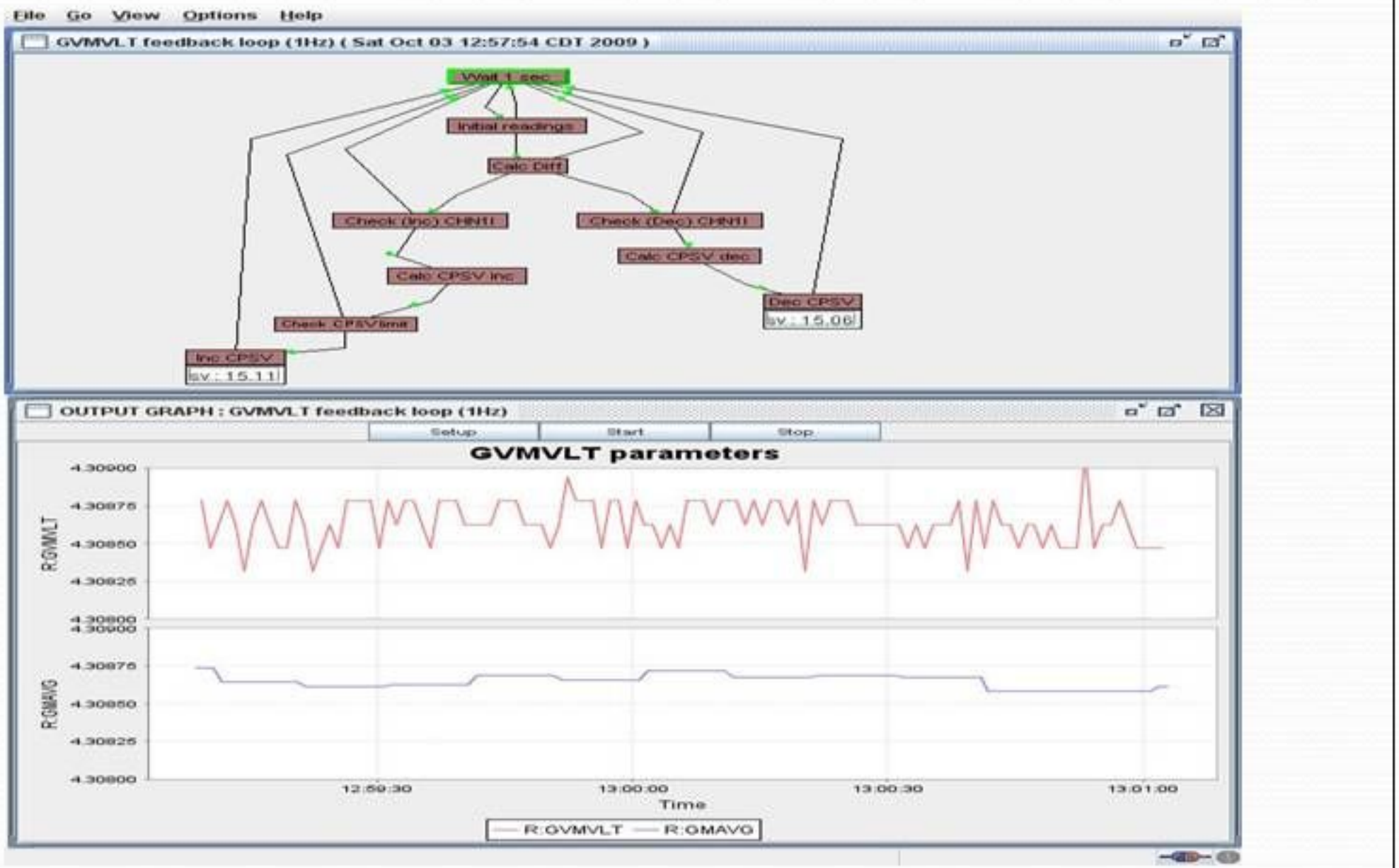
# Machine Protection System



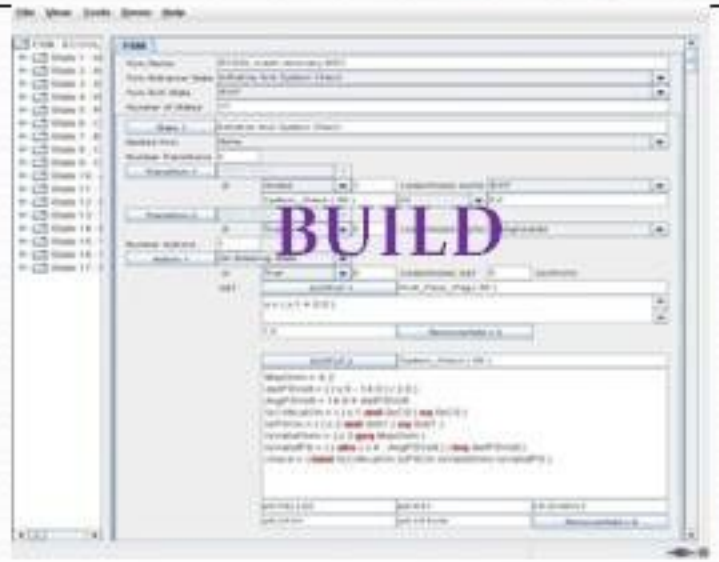
# Crash Recovery FSM

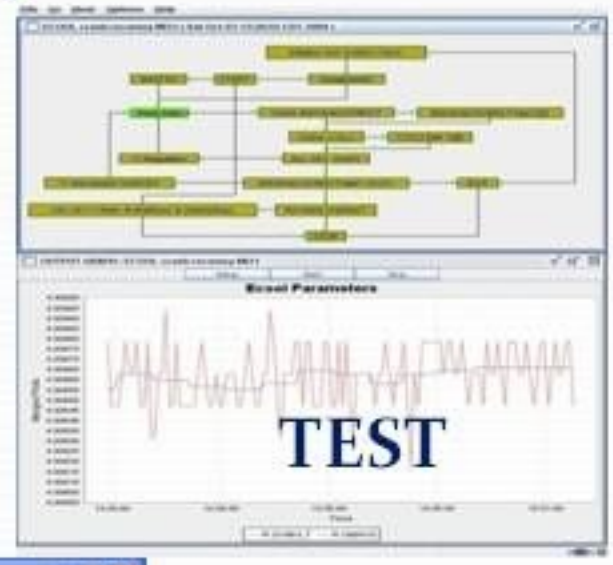


# Voltage Feedback FSM



# Development Cycle



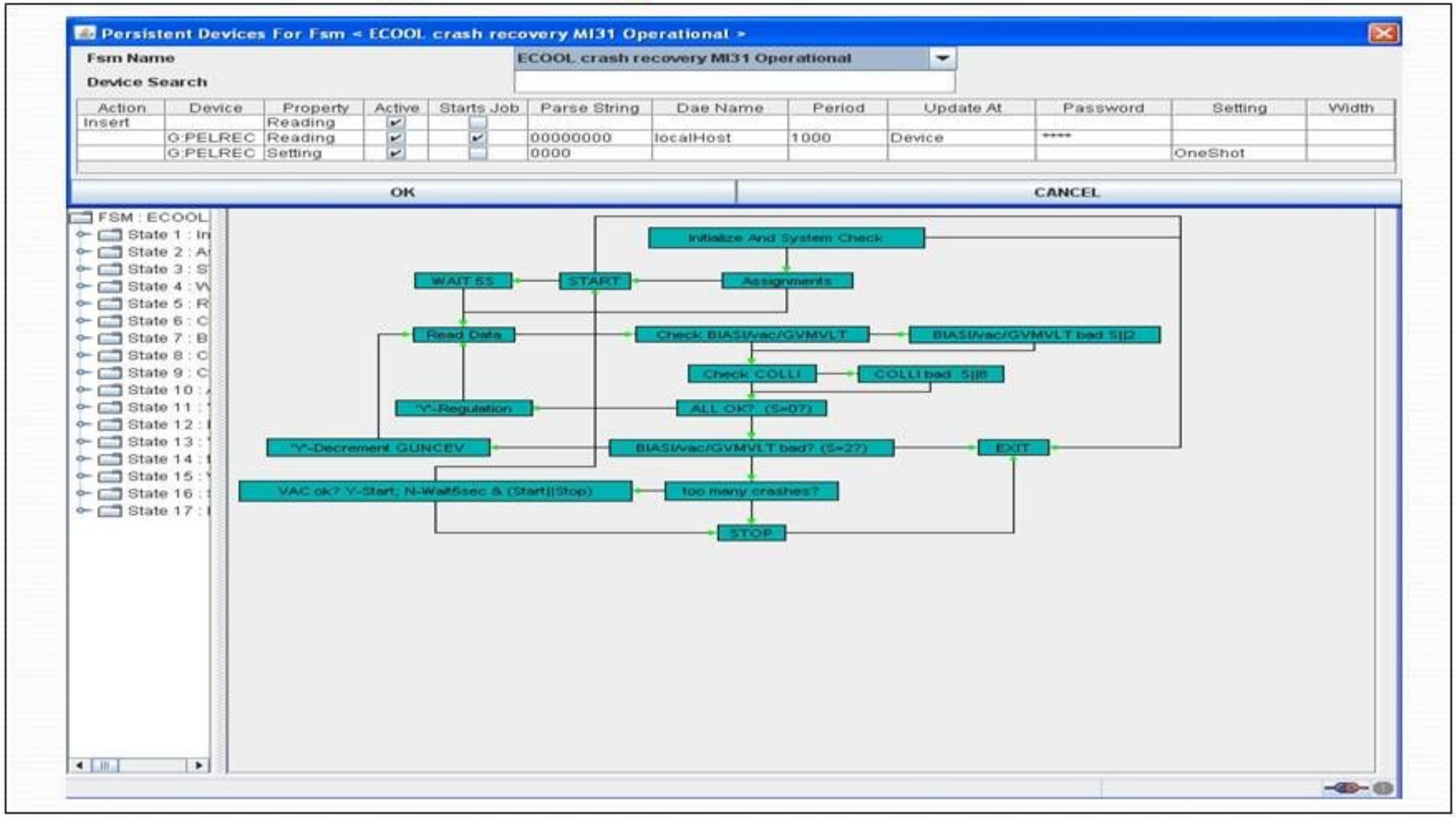

# Conclusion

- Pros
  - FSM with descriptive states provides intuitive implementation of generic tasks.
  - The hooks into the control system provide high level of power and integration.
  - The development cycle to build, test and deploy is quick and robust.
- Cons
  - Steep learning curve to get up to speed with the FSM attributes.
  - Power of multiple hooks into control system has high damage potential
- Future
  - XML-RPC server to support remote (outside firewall) users.

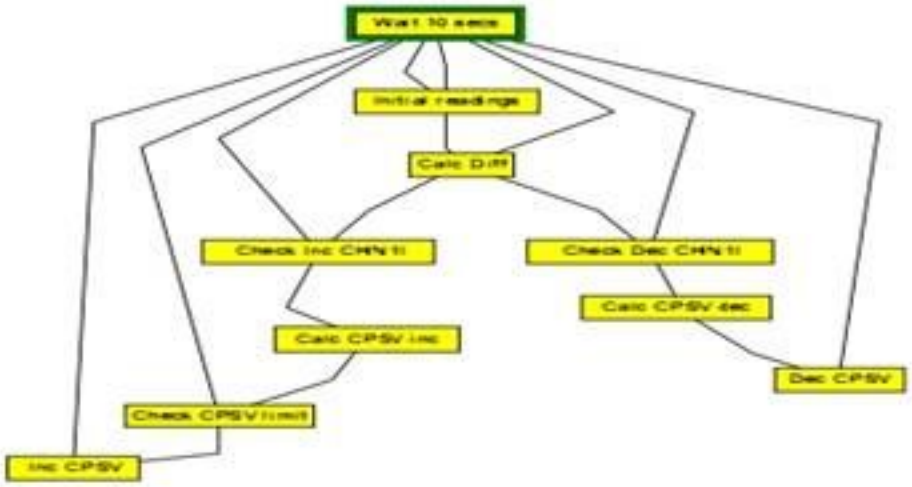
# THANK YOU FOR

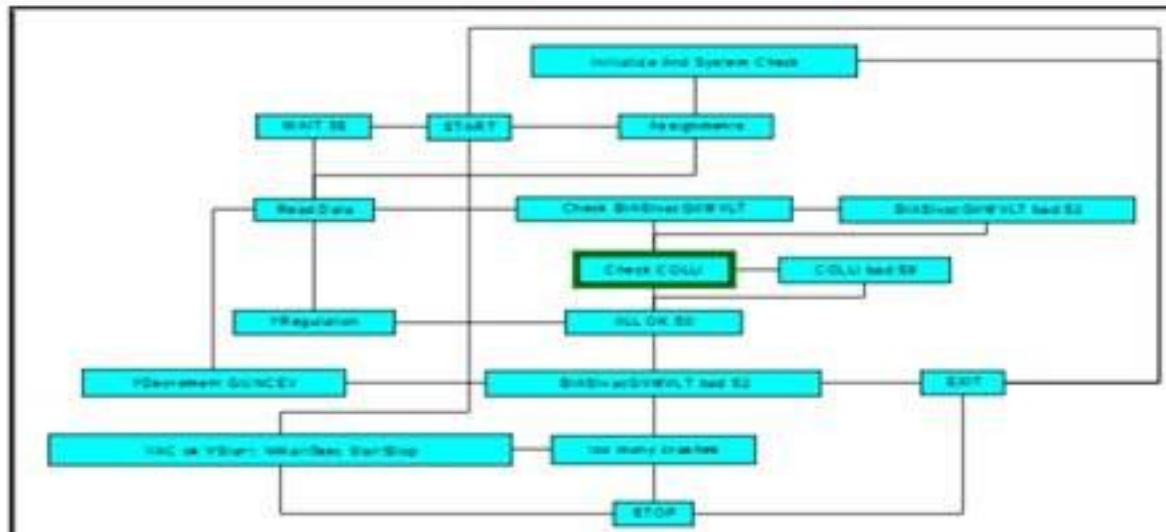
# YOUR ATTENTION

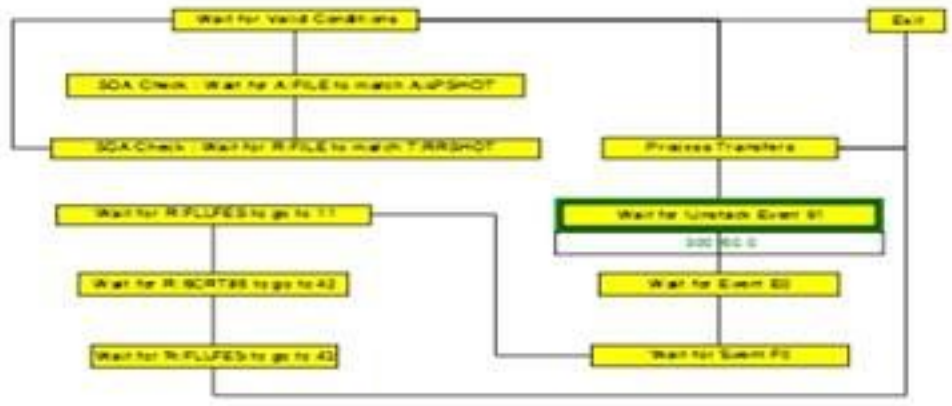
# Configuring Persistent FSM

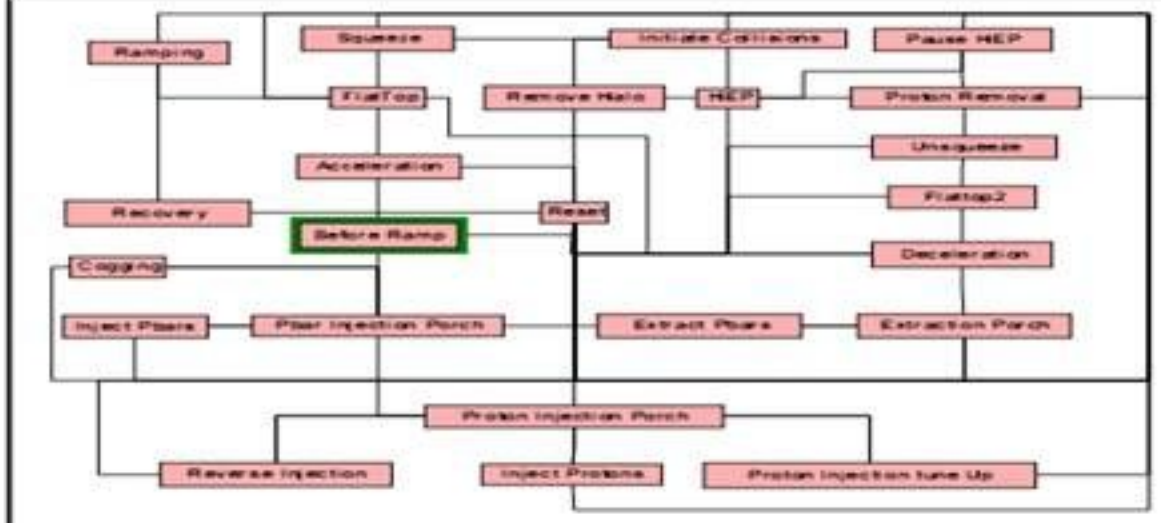


# FSM Summary (Web Page)



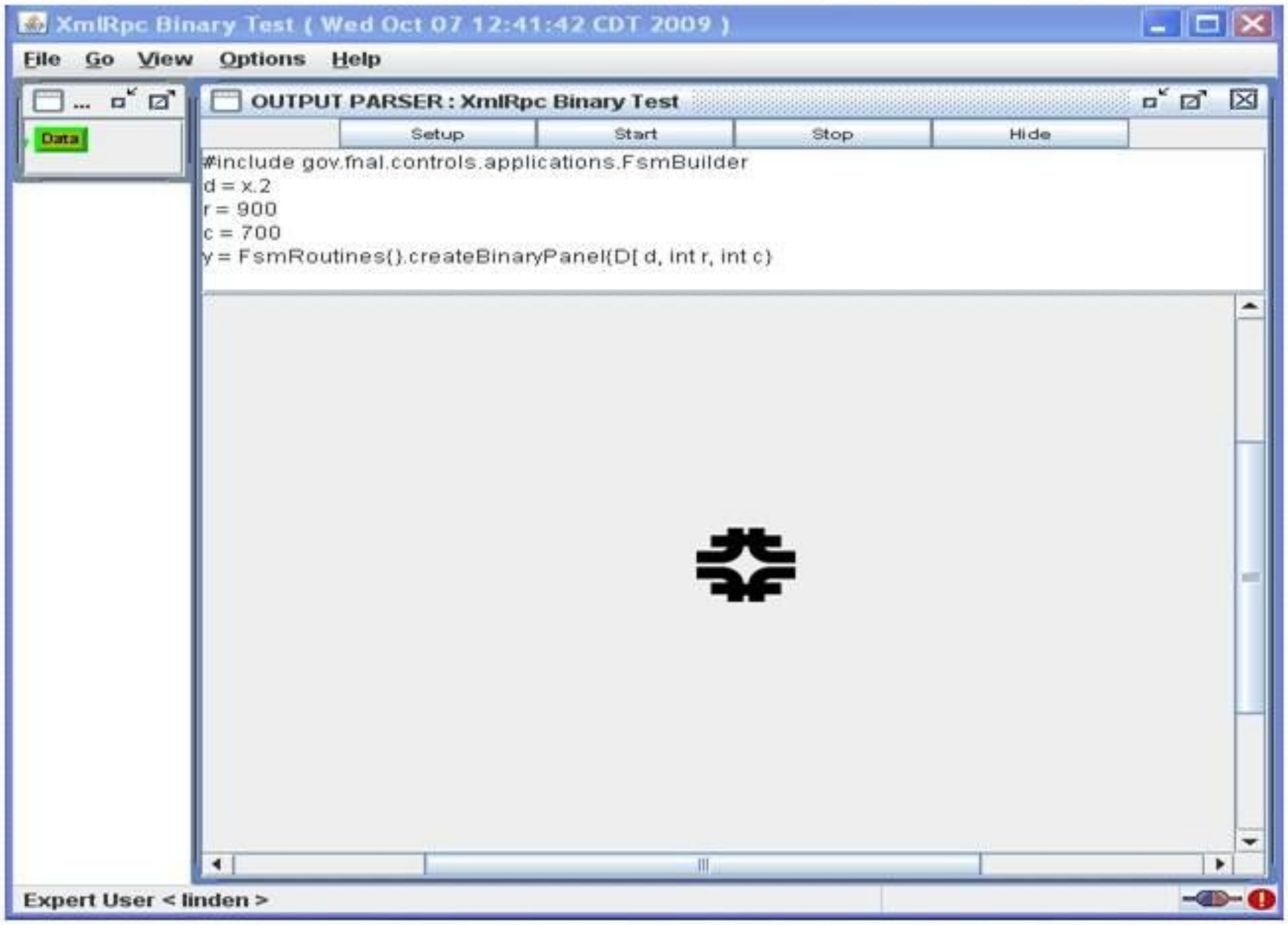



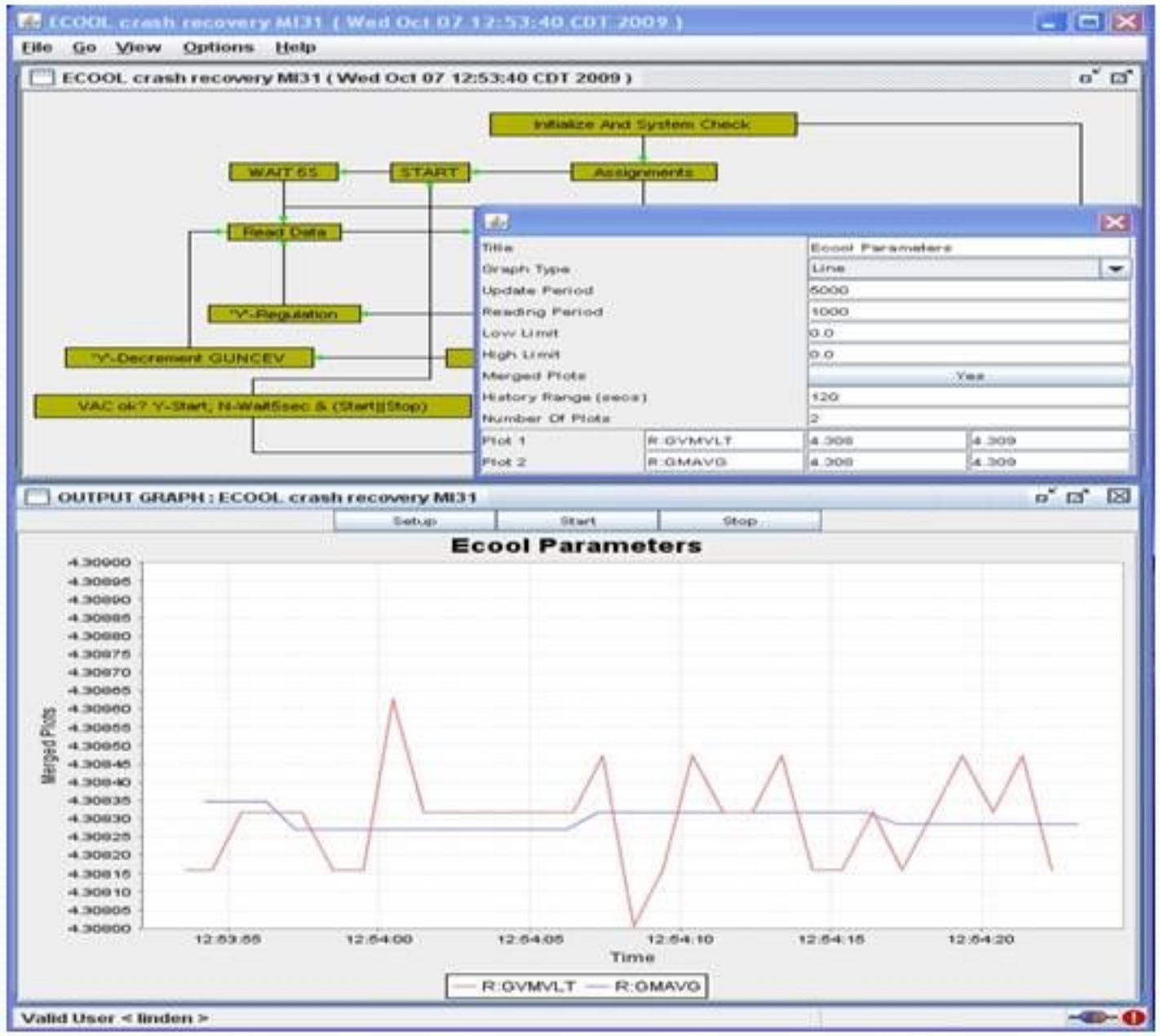
# Open Access Clients

- Processes with no user interface, always running
  - Obey same communication protocol as front-ends
  - Compare to EPICS "Soft IOC"
- Several classes:
  - Utility – Data loggers, scheduled data acquisition, virtual devices
  - Calculations – Both database driven and custom
  - Process control – Autotune of fixed target lines
  - Bridges –to ethernet connected scopes, instrumentation, control systems...
- Easy access to devices on multiple front-ends
- Friendlier programming environment than VxWorks front-ends
  - Framework transparently handles ACNET communication
- Access to database, other high level operating system features
- Do not provide hard real-time response
  - Clock events via ethernet multicast
- >100; Almost all run in Java framework; VAX framework deprecated

# FSM Views


XmlRpc Binary Test ( Wed Oct 07 12:41:42 CDT 2009 )
File Go View Options Help
Data
OUTPUT PARSER : XmlRpc Binary Test
Setup Start Stop Hide
#include gov.fnal.controls.applications.FsmBuilder
d = x.2
r = 900
c = 700
y = FsmRoutines().createBinaryPanel(D[ d, int r, int c)
Expert User < linden >

# FSM Views


ECOOL crash recovery MI31 ( Wed Oct 07 12:53:40 CDT 2009 )
File Go View Options Help
ECOOL crash recovery MI31 ( Wed Oct 07 12:53:40 CDT 2009 )
Initialize And System Check
WAIT 6S
START
Assignments
Read Data
'Y'-Regulation
'Y'-Decrement GUNCEV
VAC ok? Y-Start, N-Wait5sec & (Start||Stop)
Title
Ecool Parameters
Graph Type
Line
Update Period
5000
Reading Period
1000
Low Limit
0.0
High Limit
0.0
Merged Plots
Yes
History Range (secs)
120
Number Of Plots
2
Plot 1
R:GVMVLT
4.308
4.309
Plot 2
R:GMAVG
4.308
4.309
OUTPUT GRAPH : ECOOL crash recovery MI31
Setup
Start
Stop
Ecool Parameters
Merged Plots
Time
12:53:55
12:54:00
12:54:05
12:54:10
12:54:15
12:54:20
R:GVMVLT
R:GMAVG
Valid User < linden >